

S.B.A.C.E. GAZETTE
South Bay Atari Computer Enthusiasts

VOLUME 1
ISSUE 4
OCTOBER 1982

The material and opinions in this Gazette are those of the individual authors and do not necessarily reflect the opinions of the South Bay Atari Computer Enthusiasts. The material in this Gazette may be copied by any other User Group providing credit is given to the authors.

Please address all correspondence to:

James A. Jengo
SBACE
5025 Range Horse Lane
Rolling Hills Est., Calif. 90274

As you may already know (or perhaps not) the monthly meetings are held on the 3rd Thursday of each month at 7:00 PM at:
HW Computers
2301 Artesia Blvd.
Redondo Beach, Calif. 90278

Our next meeting will be on Thursday, October 21 st. See you there!!

Our esteemed officers are:

President	Dan Pinal
Vice President	Jerry Bransford
Secretary/Treasurer	James Jengo
Librarian	Harry Koons
Newsletter Editor	James Jengo

The SBACE GAZETTE is in no way affiliated with ATARI, Inc. ATARI is a trademark of Atari, Inc.

PHONE LIST/NEW MEMBERS

Along with this copy of our extinguished (or is that distinguished ?) newsletter you may have noticed another packet of papers that fell to the floor when you feverishly and frenziedly removed the newsletter from the envelope... it was the NEW updated telephone list containing the names, phone numbers, programming abilities and specialty interests of all our members as of the September 1982 meeting. ALSO NOTE THE COLUMN OF EXPIRATION DATES... IF YOURS IS COMING UP PLEASE RENEW YOUR MEMBERSHIP AT THE NEXT MEETING AND RECEIVE YOUR NEW MEMBERSHIP CARD AND BE CONFIDENT THAT YOU WILL CONTINUE TO RECEIVE FUTURE ISSUES OF THIS #@"#!**! NEWSLETTER.

HIGHLITES (?) OF THE SEPTEMBER MEETING

Help! Help! Help! Frank Stepczyk had to leave his post as Newsletter editor because of increasing time demands of his job (well, we certainly know where his priorities are, don't we!). Seriously, we ALL appreciate the tremendous amount of work Frank has done and extend our thanks to him. Look for contributions on Pascal and Forth, among other things, from Frank in future issues.

Why did I say "Help! Help! Help!"? Well, we need people to contribute articles to the Newsletter. There was an opulent dearth of volunteers at the last meeting!! Luckily for us, however, a few brave souls (soles ?) did volunteer: Shawn Rohan (322-2497) & Mike O'Shaughnessy will write a Game Review column (feel free to submit your game reviews to them via the Newsletter Box at HW Computers... you will get full billing for the published reviews; George Beekman will head up a column on Technical/Hardware Items; I (me) will head up the column on Assembly Language Programming. ATTENTION... heading up a column doesn't mean you are to write all the articles, just that you will screen, solicit and edit all articles on the topic. We still NEED YOU MEMBERS to submit articles. Just think, what a great way to get your name seen by and known by thousands of Atari enthusiasts and Atari, Inc. big-wigs each month! WoW, what an EGO TRIP!!!

As a further inducement (imagine a few tears shed onto my keyboard at this time) ... slight pause while I wipe off the keyboard ... HW Computers has offered to increase the usual club discount on ANY item purchased by a member whose name has been published as an author of an article which appears in the current Newsletter (you must see the store Manager to get this extra discount). In addition, Jon has also very kindly offered the use of any of the Store's word processors (even the B I G G G ones) for us to type up these works of art. Thank you very much, Jon!

Also (still desparate to get articles) the club voted to award a prize in the form of a \$25.00 gift certificate to the best overall article of each two Newsletters (for this month though, the prize will be for the best article in this Newsletter... don't you wish you had submitted that article?). The selection of the "best" article will be by majority vote of the members present at the meeting following publication.

The idea was brought up of having a basic level BASIC programming class starting 1/2 hour before each meeting. This was well received and we even had a volunteer to teach this basic BASIC course. Announcements will be made at the upcoming meeting, with the course scheduled to start with the November meeting.

We all agreed that we would like to see columns dedicated to various programming languages including BASIC, FORTH, PILOT and PASCAL (ASSEMBLY is already in this issue). We need volunteers and writers!!!

Dan Pinal gave an interesting talk on "loop programming" using Assembly language. Please think of topics you would like to have speakers speak on in the future, and let us know at the meeting.

NEWS FROM THE LIBRARIAN

A recent addition is a quarterly subscription to the Microcomputer Index, a listing of current articles and their sources (from over 20 computer magazines) on all aspects of microcomputers.

Another addition has been a Club exchange subscription to ANTIC magazine. The publishers of ANTIC kindly supply our club with the subscription in exchange for a subscription to our Newsletter... THANK YOU ANTIC.

Also, through the generosity of the Valley Users Group we have swapped diskette program libraries. Harry has these in the disk library already. By the way, Harry Koons deserves a lot of praise for the great job he has (and is) done in updating our diskette library (ed.).

NEW BUSINESS

Atari, Inc. has asked all Users' Groups to re-register with their Users' Group Support Section. This consists of sending them the list of current officers, and the membership roster (which they promise will not be released to outside agencies). We sent our data back to Atari the day after the last meeting. Atari says that they intend to increase interaction and support of Users' Groups in many ways, including supplying video tapes for club meetings... THANK YOU ATARI. (Haven't heard anything from them yet!?!)

One of our members has made the big time by having a program review article published in issue #33 (the current one) of SOFTSIDE (on p.74). CONGRATULATIONS to Rick NICHOLS (incidentally, Rick was first inspired by writing reviews for our Newsletter). Another of Rick's reviews will be published in the near future in a different publication. Keep up the good work!

Another of our members (James Jengo) has written a totally machine language program which is being marketed by COSMI, Inc. under the name: *Crypts of Plumbous*. Rumor has it that he is considering moving to Easy Street and getting another couple yachts! (He informs us that autographs will be available, by appointment only of course.)

GETTING TO KNOW (AND LOVE) ASSEMBLY LANGUAGE

by James A. Jengo

You've got to be kidding you say. Well, not really, I say. What assembly language lacks in slick commands it more than makes up for in utility, memory conservation and S P E E D !!! Since this is THE language that the computer "uses", you obviously (maybe not at this point) can execute any program commands in assembly that you wrote in ANY other language that is supported on your computer. Be patient though.

Starting with this issue I will be writing a series of columns on basic (no pun intended) assembly language concepts and programming. Hopefully, even if you have no desire to actually program in assembly, you will at least have a feeling for what the language is like and, via some of the demo programs, will be awe-stricken with its power. Now that's an awesome (or is that awful) statement to make.

The program listings will be commented and will work if typed in using any Atari-compatible assembler/editor. For the demo programs I will also list the program in numerical form (via Data statements in BASIC) so that those of you without assembler/editors will be able to be devastated with the awesome power of assembly language whenever the desire stirs within you.

Well, to work. Assembly "is" a very logical language (it's the perfect match for the computer). It only does ONE thing at a time. Therefore it may take more commands to execute a simple chore, but it'll be worth it (unfortunately this does make it more difficult to debug, but the solution to that problem is simple: simply do not make any errors). By the way, the examples I will give will not use the most efficient code to accomplish their task. My main goal is to make the process very understandable (also I don't want to be embarrassed by writing inefficient code, thus this excuse).

Assembly language offers you three working areas called registers (the A register, the X register and the Y register). You may place numbers in these registers, perform arithmetic operations on them, compare them, increment them, decrement them and make logical (don't you wish) decisions based on their values, and even store the contents of these registers anywhere you want in memory. Just as in BASIC you may make up names for your variables (only 6 letters long though) but you must go to the slightly extra trouble of telling the computer what addresses in memory these variables will represent (BASIC does that automatically; however, this way, you can do a lot of tricky things (really powerful isn't it?)). Depending on the results of various operations you can check FLAGS (Carry flag, Zero flag, etc.) and decide to alter memory values, jump someplace else in your program, etc. (kind of like the "if...then" statement in BASIC).

Let's keep going. The hardest thing I had to get used to was the fact that you can only do all these operations on single byte numbers, i.e. all numbers used must have values between 0 and 255. If you think about it for awhile you'll realize that just about any value number you might commonly use can be represented by 2 bytes... so that's what we do: we use two bytes (high and low byte) to represent each number. Don't get scared, you'll catch on pretty fast (otherwise you might consider burning this article). But, you say, 2 bytes can only represent numbers with values up to around 64000, and YOU want to use bigger numbers. Well, you may either divide these numbers by some factor, do your thing and then multiply the result by that factor; or you may use three (or more) bytes to represent your number so that you can deal with very big (or very small) numbers.

What exactly do you mean by "low byte" and "high byte" you may ask. A fairly simple way of looking at this is that the high byte is simply a counter that is incremented by 1 each time the value of the low byte exceeds the value of 255. At that point the high byte is incremented by 1 and the low byte is reset to '0' and the remaining values (if any) added to the low byte. (What did he say??) For example, if the high byte had a value of 2 and the low byte had a value of 250, this pair of bytes is a representation of the number 762 ($2 \times 256 + 250 = 762$). Now, if we add 8 to the number (762) stored as a two byte combo, the computer would first try to add the 8 to the low byte (250) resulting in the number 258. However, since this is greater than the maximum possible value any byte can have (255), an overflow FLAG is set and the value of the low byte is reset at what's left over from the 8 after enough units have been added to the low byte to = 256 (in our example the low byte originally was 250 so 6 units of the 8 we wanted to add would make the low byte = 256, thus leaving 2 units left, which is the value the low byte would now be set to). In our addition program we would check to see if the overflow flag (called the *Carry Flag*) is set, and if so (it would be in our example since the low byte reached 255 and recycled) we would increment the high byte by 1 thus giving it a value of 3. In summary, if we start out with a high byte of 2 and a low byte of 250, then add 8 to the number represented by this two-byte combo, the new combo would have the high byte = 3 and the new low byte = 2 (thus the answer is $3 \times 256 + 2 = 770$).

Whew, pretty easy, eh?!

Another way of looking at this is that to calculate the number represented by a two-byte combo we multiply the value stored in the high byte by 256 and add the value stored in the low byte to it. In order to convert a 'regular' number to a two-byte combo, we divide the number by 256 and store the integer result (i.e. ignore any fractional remainder) in the high byte, then subtract the product of this new high byte value * 256 from our original number, and store the answer (the difference) in the low byte.

If this isn't all crystal clear right now don't worry. It's a new concept to learn. Just go back and reread the information and the example and soon it'll start to make sense (hopefully). Otherwise just send me \$500,000. in negotiable stocks and I'll be happy to consider a personal session at reasonable hourly rates!?!

Let's look at some real programs. These programs will demonstrate how to add two numbers together using an assembly subroutine which is called from a BASIC main program. The first listing is the simplest, assuming that each of the 2 numbers to add are <255 and that the resultant sum is <256. This program best shows off the logic of addition in assembly language. The second listing does the same thing but will handle numbers of up to around 64,000 (much more practical). Incidentally, we are only talking about integer addition.

```

10  *= $600 ;tell the computer that the program is starting at
      address $600 (1536 decimal)
20  FIRST = $420 ;reserve memory location $420 (scratch area
      in the cassette buffer)
30  SECOND = $421
40  PLA ; always use this (will explain in the future)
50  PLA
60  PLA ;transfer the first number from the BASIC program
      into the A register
70  STA FIRST ;store the contents of the A register
      (i.e. the FIRST number from BASIC) in the
      memory location we called FIRST

```

```

80  PLA
90  PLA ;transfer the second value (second number) from BASIC
      to the A register
100  STA SECOND ;store the contents of the A register in the
      memory location you called SECOND
110  LDA FIRST ; we're now ready to start the addition. We first
      load the A register with the value of the first
      number
120  CLC ; we must always Clear the Carry flag before an addition
130  ADC SECOND ;now we add the value stored at the memory location
      SECOND to the contents of the A register. The sum
      will be in the A register.
140  STA $D4 ; the address to store the low byte of the answer in
      order to pass it back to the calling BASIC program.
150  LDA #0 ;we set the high byte = 0 since we stipulated that
      the 2 numbers to be added and the sum would be <255.
160  STA $D5 ;the address to store the high byte of the answer in
      order to pass it back to the calling BASIC program.
170  RTS ;code to tell the program to Return To Subroutine, i.e.
      to return control back to the calling BASIC program.
180  .END ;tells assembler that this is the end of the program.

```

Holy Smokes, it's 2:14 AM!!! Therefore I think I will postpone the more complete (high and low byte) version of this program until next Newsletter. Below is a sample BASIC program that will ask for two numbers to be added together, call the assembly subroutine to add them together and then display the resultant sum. It, and the above routine, should work.

```

10 ? "Enter first number (<128) ";;INPUT FIRSTNUM
20 ? "Enter second number (<128) ";;INPUT SECONDNUM
30 SUM=USR(1536,FIRSTNUM,SECONDNUM)
40 ? "Sum = ";SUM
50 END

```

It's time to go now. Please let me know if the above was too simple, too difficult or too confusing. Also, recall that we do need contributors, so please submit articles! Any and all input will be greatly appreciated.

By the way, in case you're interested, the entire Newsletter was printed using the Atari Word Processor in conjunction with an Epson MX-80 printer with GRAFTRAX^{PLUS}. The phone list was made using the Filemanager 800 program.

See you all next issue.

ATARI USERS GROUP SPECIALS FROM HW ELECTRONICS

for the month of
October 1982

- 1) Percom Master Drive for the Atari
- Operation in either single density (88K) or double density (170K)
 - Single or dual head capability
 - First drive contains four drive controller - you save on the cost of additional drive units

In order to run the Percom Disk Drive you must have DOS 2.0S (Atari)

RED40-S1 MASTER DISK DRIVE
Cat. No. 3815 Atari, 24K \$ 595.00

- 2) 32K Ramcram for Atari
- Memory expansion for the 400 and 800 - 400 requires installation

Cat. No. 3003 \$ 99.95

- 3) Okidata Microline 82A Printer
- 80 columns (132 condensed)
 - 120 cps
 - bi-directional
 - logic seeking
 - parallel interface

Cat. No. 4200 \$ 449.00

- 4) Verbatim 5 1/4" Diskettes

Cat. No. 1147 Soft sector, Reg. box \$ 23.95
SD/DD - Datalife

- 5) Flip 'n File 5 1/4" Disk Protector Cases

Cat. No. 2956 Holds 50 diskettes \$ 19.95

- 6) Printer Paper

Cat. No. 1456 30 lb., 9 1/2" w \$ 24.95
1/2" sprocket

